ITC 4/52
Information Technology and Control
Vol. 52 / No. 4 / 2023
pp. 1058-1072
DOI 10.5755/j01.itc.52.4.34021

Few-shot Sentiment Analysis Based on Adaptive Prompt Learning and Contrastive Learning

Received 2023/05/04 | Accepted after revision 2023/07/25

HOW TO CITE: Shi, C., Zhai, R., Song, Y., Yu, J., Li, H., Wang, Y., Wang, L. (2023). Few-shot Sentiment Analysis Based on Adaptive Prompt Learning and Contrastive Learning. *Information Technology and Control*, 52(4), 1058-1072. https://doi.org/10.5755/j01.itc.52.4.34021

Few-shot Sentiment Analysis Based on Adaptive Prompt Learning and Contrastive Learning

Cong Shi, Rui Zhai, Yalin Song, Junyang Yu, Han Li, Yingqi Wang, Longge Wang
School of Software, Henan University, Kaifeng 475004, China; Henan Provincial Engineering Research Center of Intelligent Data Processing, Henan University, Kaifeng 475004, China

Corresponding author: zr@henu.edu.cn

Traditional deep learning-based strategies for sentiment analysis rely heavily on large-scale labeled datasets for model training, but these methods become less effective when dealing with small-scale datasets. Fine-tuning large pre-trained models on small datasets is currently the most commonly adopted approach to tackle this issue. Recently, prompt-based learning has gained significant attention as a promising research area. Although prompt-based learning has the potential to address data scarcity problems by utilizing prompts to reformulate downstream tasks, the current prompt-based methods for few-shot sentiment analysis are still considered inefficient. To tackle this challenge, an adaptive prompt-based learning method is proposed, which includes two aspects. Firstly, an adaptive prompting construction strategy is proposed, which can capture the semantic information of texts by utilizing a dot-product attention structure, improving the quality of the prompt templates. Secondly, contrastive learning is applied to the implicit word vectors obtained twice during the training stage to alleviate over-fitting in few-shot learning processes. This improves the generalization ability of the model by achieving data enhancement while keeping the semantic information of input sentences unchanged. Experimental results on the ERPSTMT datasets of FewCLUE demonstrate that the proposed method have great ability to construct suitable adaptive prompts and outperforms the state-of-the-art baselines.

KEYWORDS: Few-shot Sentiment Analysis; Adaptive Prompt Learning; Contrastive Learning; Dot-Product Attention; Semantic information of tests.

1. Introduction

Sentiment analysis mainly involves the use of deep learning strategies and natural language processing (NLP) to identify, extract, and analyze the emotional tone of texts [21]. It is a research focus in the fields of NLP and has applications in numerous areas, including social media, customer feedback, and political analysis. However, traditional sentiment analysis techniques face significant challenges when confronted with small-scale datasets. To overcome this challenge, few-shot learning has emerged as a viable solution for handling few-shot sentiment analysis tasks. In recent years, two categories of approaches have been proposed, namely the "pre-training-fine-tuning" and "pre-training-prompt-prediction" paradigms.

The "pre-training-prompt-prediction" paradigm has gained more attention than the "pre-training-fine-tuning" paradigm in recent years. It involves obtaining a pre-trained model through unsupervised training on large-scale data using large models like BERT [4]. The model is then fine-tuned on small sample datasets and makes predictions using prompt learning methods. The construction of prompt templates is the key of this approach since an appropriate prompt definitely improves the performance of the model. However, the existing hand-crafted prompt template construction method, PET [24], and the automatic prompt template construction method, P-tuning [19], do not fully utilize the semantic information of the input sentences during the process of template construction. Additionally, the prediction accuracy of different templates vary greatly in the hand-crafted prompt template construction method, making it challenging to find the "best" prompt template. Thus, the current research challenge is how to automatically construct templates by fully utilizing the semantic information of the input text.

Few-shot learning [1, 8, 25, 26, 6] involves the use of a small quantity of labeled data for model training. This approach transfers the knowledge the model has learned from massive quantities of unlabeled data to less sample data. Contrastive learning is often used in few-shot learning tasks to train the model by comparing the similarity between two samples and learning a better representation. It minimizes the distance between samples of the same class and maximizes the distance between samples of different classes. Regularization techniques have a great impact on preventing over-fitting and improving the generalization ability of the deep learning models, especially in few-shot learning tasks. While Dropout is a commonly used regularization method, Liang et al. [14] have pointed out that a nonnegligible inconsistency exists between the training and inference stages of Dropout. That is, the randomly sampled submodel (caused by Dropout) obtained in the training stage is inconsistent with the full model (without Dropout) in the inference stage.

To address the aforementioned problems, we propose a few-shot sentiment analysis method that leverages adaptive prompt learning and contrastive learning. Specifically, we design an adaptive prompt module to automatically create prompt templates based on the semantic information of texts. It enables the model to focus on the most informative features of the input sequences. In addition, we apply a method of contrastive learning to enhance the performance of sentiment analysis on small sample data and improve the generalization ability of our model. The contributions of this paper are shown as follows:

1 We propose a method based on few-shot learning for sentiment analysis, called Adaptive Prompt with R-Drop (APRD), which utilizes an attention mechanism and is particularly effective in low-resource scenarios. Furthermore, we demonstrate that pre-training on existing labeled datasets from diverse domains can significantly enhance the ability of prompt template construction, leading to improved model performance.

2 We employ the R-Drop contrastive learning method in the inference stage to enhance the robustness of the model against dropout. R-Drop adds a regularization term to the model, making the outputs consistent under different dropout rates, which enhances the similarity of "model averaging" and "weight averaging". As a result, this approach alleviates over-fitting and improves the generalization ability of our model.

The other parts of this article are organized as follows: First, in Section 2, we express a summary of the main related works in areas of few-shot learning and sentiment analysis. Next, in Section 3, we provide a detailed introduction to proposed method, which focuses on the mechanisms, mathematical analysis, and training process of the adaptive prompt module and the contrastive learning module. Then, in Section 4, we present the results of our experiments along with related analyses. Finally, in Section 5, we provide a summary of our article and discuss our further research directions.

2. Related Work

Researches on sentiment analysis has become prevalent in NLP since 2000 [29]. To date, it has mainly undergone two major phases: the period of traditional sentiment analysis and that of the deep learning-based sentiment analysis.

2.1. Period of Traditional Sentiment Analysis

The period of traditional sentiment analysis can be divided into two stages, which are discussed in the following subsections.

2.1.1. The Stage of 'Emotional Dictionary and Rules'

Such methods perform sentimental tendency classification of text sentences by manually constructing a sentiment dictionary and applying a dictionary and rules-based approach to it. Specifically, they utilize the dictionary to get the emotion value of the emotional words in the text, and then try to determine the overall sentiment tendency of the text by conducting a weighted calculation on them [3, 20]. However, these methods tend to ignore the association between words, leading to an unchangeable emotional value of words in different scenarios and contexts. Therefore, the establishment of the relevant sentiment dictionaries that target specific scenarios to raise the accuracy rate of classification is necessary. Due to the flexibility of languages, constructing a sentiment dictionary of both generosity and high-quality remains challenging.

2.1.2. The 'Stage of Feature Engineering'

During this stage, task-specific models are trained solely on input-output sample datasets of the target task. N-gram model is used in combination with classical machine learning classifiers such as K-Nearest Neighbor (KNN), Naive Bayes, Maximum Entropy, and SVM to perform supervised learning [11]. However, these methods are not able to take into account the relationship between variables and have poor generalization performance. Li et al. built a model with a priori knowledge of categorized information using Term Frequency-Inverse Document Frequency (TF-IDF) to extract the most meaningful features from unstructured texts [15]. This algorithm is easy to implement and understand but relies heavily on the corpus and has low precision. To sum up, the shortcomings of traditional machine learning-based sentiment analysis are apparent: it requires a high-quality corpus that matches the processed text for training, resulting in poor algorithmic accuracy.

2.2. Deep Learning-based Sentiment Analysis

Classical neural networks such as Recursive Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Bi-LSTM networks have all been applied in sentiment analysis tasks [5]. Li et al. proposed a stacking structure called BCSA of Bi-LSTM and CNN to improve the ability of the model to recognize sentiment [14]. Chen et al. proposed HUSN, which enhance hierarchical neural networks on basis of users' reviewing habits [2]. Sadr et al. constructed a model that employs RNN, making use of its tree structure as a substitute for the pooling layer in the convolutional network, for the purpose of capturing long-term dependencies and reducing the loss of local information [23]. Deep learning-based methods have made marvelous progress compared with traditional methods. However, these algorithms are not applicable when faced with small amounts of labeled data. Additionally, significant innovations made in the NLP field have been promptly applied in sentiment analysis tasks. Mikolov et al. proposed the Continuous Bag of Words (CBOW) model combined with Skip-Gram model, which uses contextual words to predict target words and uses the target words to predict the surrounding words of the text [13]. Sun et al. used GloVe to train word vectors and Bi-directional Gated Recurrent Unit (BGRU) to obtain contextual information. They also used an attention mechanism to process sparse data, which was confirmed to be effective on the IMDB dataset [27]. However, Word-2Vec and GloVe obtain static word vectors, which ignore the relationship of the word context, leading to the failure of the model in capturing advanced semantic information from the text.

As the role of fully-supervised learning becomes increasingly smaller, recent research findings in the field of NLP have been applied in sentiment analysis tasks, including "pre-training-fine-tuning" paradigm and the burgeoning "pre-training-prompt-prediction" paradigm.

2.2.1. "Pre-training-fine-tuning" Paradigm

This paradigm involves pre-training models with fixed architectures on large-scale unlabeled data to predict the probability of observed textual data as a language model (LM), which is then fine-tuned using a small quantity of manually labeled data in downstream tasks. Pre-trained language models (PLMs) aim to train models on large quantities of corpus to enable them to learn the probability distributions for every single word in the corpus, building models that fit these textual distributions. The contextual problem of ignoring contextual connections was solved by the proposal of dynamic word vector algorithms ELMo [22] and BERT [4]. To train word vectors, Liu et al. [18] and Fang et al. [7] used BERT as a pre-trained language model instead of Word2Vec and GloVe, resulting in better classification results when embedded into other models. Sun et al. [27] used a deep belief network (DBN) to solve the problem of sparse text features, while Heikal et al. [9] defined an integrated model from the best CNN model and Bi-LSTM model, which greatly improved classification accuracy. However, all the methods mentioned above have high requirements on fine-tuning strategies of models, together with the parameters of the model and complexity, which are difficult to control.

2.2.2. "Pre-training-prompt-prediction" Paradigm

With the rising of prompt learning, the "Pre-training-prompt-prediction" paradigm has become a new research hotspot. This approach is characterized by training a single LM in a completely unsupervised manner to solve a large number of tasks after providing a suitable set of prompts. Schick et al. proposed the training strategy Pattern Exploiting Training (PET) for semi-supervised tasks [8]. In this strategy, the input sample is redefined as cloze sentences under the aim of helping the language model constuct the given task. This is the first strategy of prompt learning. Prompt learning requires the design of a prompt template, which can be obtained through manual design [5]. Hu et al. proposed the KPT method based on PET, which extended and improved the labeled words of PET by introducing a knowledge base to it [10]. Ye et al. proposed an ontology-enhanced knowledge prompt method, short for OntoPrompt. In this strategy, external knowledge is implanted into the framework of prompt learning in text form to realize the model's perception of tasks and domains [31]. From the above works, we can see that the quality of the prompt templates has strong influence on the performance of downstream tasks. Therefore, it is essential to find the most appropriate prompt in prompt learning methods. Liu et al. [24] proposed the P-tuning method. It achieved the automatic construction of the template by using tokens never seen in the model to form the prompt, which transforming the template construction problem into a continuous parameter optimization problem. Although these methods improved the defect of PET of strong subjectivity and small coverage to some extent, they still failed to fully exploit the semantic information of the input text in the process of rapidly building the prompt templates.

Utilizing a pre-trained LM for sentiment analysis tasks through prompt learning is a challenging task. Firstly, it is not easy to find suitable prompt templates, as they require specific design considerations. Secondly, using a prompt learning method with only a single prompt to guide a pre-trained LM to complete a sentiment analysis task may be a suboptimal approach, as the results may differ significantly from the pre-trained targets.

3. Structure of the Model and the Whole Training Process

This paper introduces an Adaptive Prompt Template Construction Method (APRD) that combines contrastive learning with dot-product attention mechanisms. The model comprises three parts: the Encoding and Embedding module, the Adaptive Prompt Learning module, and the Contrastive Learning module, as illustrated in Figure 1. The Adaptive Prompt Learning module aims to dynamically construct templates based on input sequences, while the Contrastive Learning module amplifies data and mitigates overfitting.

Figure 1

Framework of the whole model

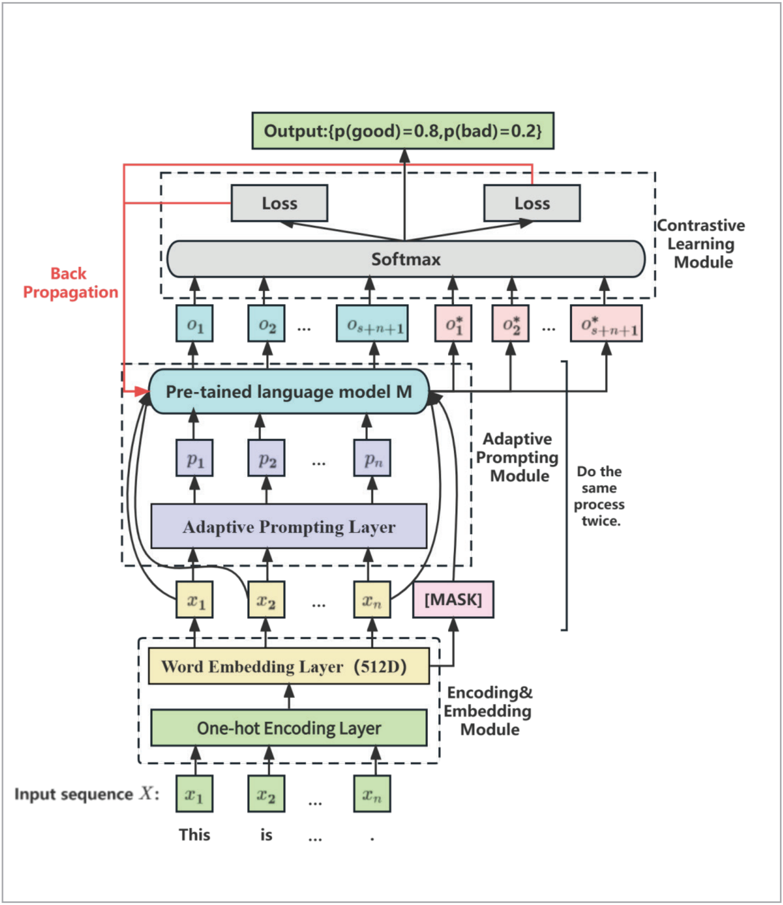
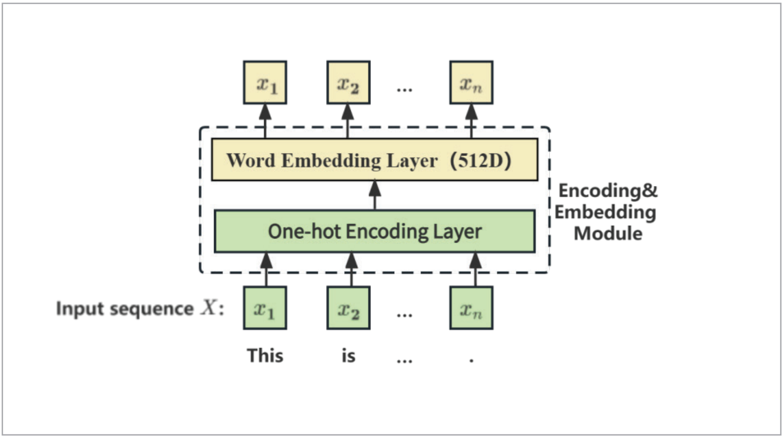

Output:{p(good)=0.8,p(bad)=0.2}
Loss
Loss
Softmax
Contrastive Learning Module
Back Propagation
o_1 o_2 ... o_{s+n+1} o_1^* o_2^* ... o_{s+n+1}^*
Pre-tained language model M
p_1 p_2 ... p_n
Adaptive Prompting Module
Do the same process twice.
Adaptive Prompting Layer
x_1 x_2 ... x_n [MASK]
Word Embedding Layer (512D)
One-hot Encoding Layer
Encoding& Embedding Module
Input sequence X:
This is ... .

Figure 2

Structure of the Encoding and Embedding module



3.1. Encoding and Embedding Module

This part comprises two layers: the One-Hot Encoding layer and the Word Embedding layer. Its purpose is to encode and generate word embeddings for each word $\{x_1, x_2 \ldots x_n\}$ in the input sequence X, transforming them into 512-dimensional vectors. The resulting set of vectors is denoted as $\{x_1, x_2 \ldots x_n\}$, as shown in Figure 2.

3.2. Adaptive Prompt Learning Module

The idea for improvement in this module is based on the traditional hand-crafted prompt learning method (HPL). Therefore, we first introduce the HPL model. The structure of the model is shown in Figure 3.

Suppose there is a pre-trained model *M*, and the input sequence is a sentence *X* that have already been tokenized. Let *n* stands for the length of the sentence *X*, and the dimension of the word vector in word embedding layer is 512. By sending X to the one-hot encoding layer and the word embedding layer, a set of 512-dimensional vectors with a length of *n* is obtained, which is denoted as $\{x_1, x_2 \ldots x_n\}$. Next, a hand-crafted prompt template is denoted as $\{p_1, p_2 \ldots p_i, [MASK], p_{i+1} \ldots p_m\}$, where $[MASK]$ is a 512-dimensional vector obtained after being converted by the word embedding matrix. Then, we use the set of 512-dimensional vectors of $\{p_1, p_2 \ldots p_i, [MASK], p_{i+1} \ldots p_m\}$ and $\{x_1, x_2 \ldots x_n\}$ as the input sequence of the pre-trained language model M (such as BERT), from which a set of 512-dimensional vectors with a length of *m* + *n* is obtained, denoted as $\{o_1, o_2 \ldots o_{m+n}\}$.

Finally, the output $\{o_1, o_2 \ldots o_{m+n}\}$ of the pre-trained language model is utilized to calculate the probability of the location of the $[MASK]$ in the input template. The word with the highest probability in the word list is then selected as the "best word". For instance, to perform sentiment analysis on the sentence "The weather is good", a manually-crafted prompt template such as "It is very $[MASK]$ today" can be used. Assuming that the language expresser is set as {"good", "bad"}, the manually-crafted prompt model combines the constructed template and the original text as "It is very $[MASK]$ today. The weather is good." Ultimately, the pre-trained language model M returns the predicted value of the sentence.

Compared to the manually-crafted prompt template method in Figure 3, our adaptive prompt learning method can automatically create prompt by the adaptive prompt layer. The whole structure of the adaptive prompt module is shown in Figure 4.

In this module, we adopt the seq2seq structure based on the dot-product attention mechanism as the adaptive prompt layer. The max prompt length is set to 300. It is composed of an encoder, a decoder, and the dot-product attention structure. As shown in the left part $\{x_1, x_2 \ldots x_n\}$ of Figure 5, the encoder encodes the input sentences word by word, yielding a set of 512-dimensional vectors of length $n: \{h_{1,k}, h_{2,k} \ldots h_{n,k}\}$. The function of the structure that links the encoder and decoder is to apply the attention structure to use the sequence $\{h_{1,k}, h_{2,k} \ldots h_{n,k}\}$ to generate

Figure 3
Whole model structure of the traditional hand-crafted prompt learning method (HPL)

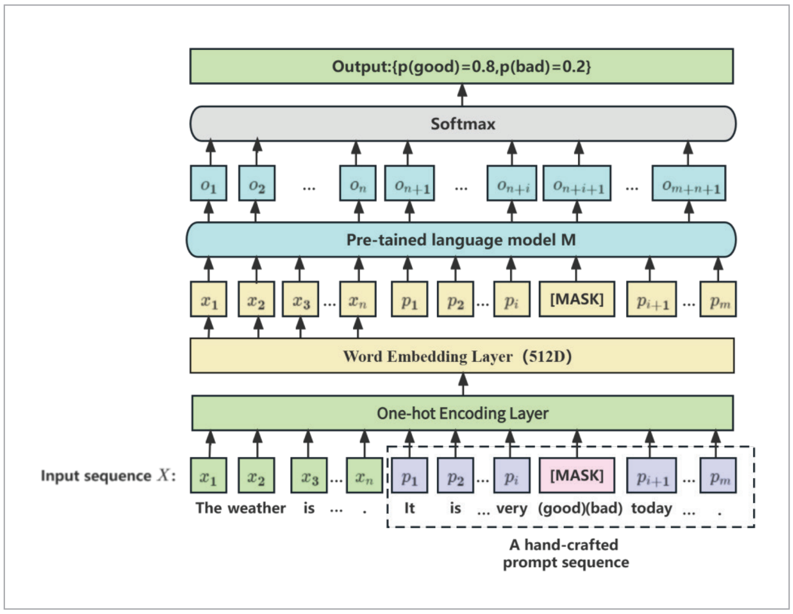

Output:{p(good)=0.8,p(bad)=0.2}
Softmax
o_1 o_2 ... o_n o_{n+1} ... o_{n+i} o_{n+i+1} ... o_{m+n+1}
Pre-tained language model M
x_1 x_2 x_3 ... x_n p_1 p_2 ... p_i [MASK] p_{i+1} ... p_m
Word Embedding Layer (512D)
One-hot Encoding Layer
Input sequence X:
The weather is ... . It is ... very (good)(bad) today ... .
A hand-crafted prompt sequence

Figure 4
Whole structure of the adaptive prompt learning module

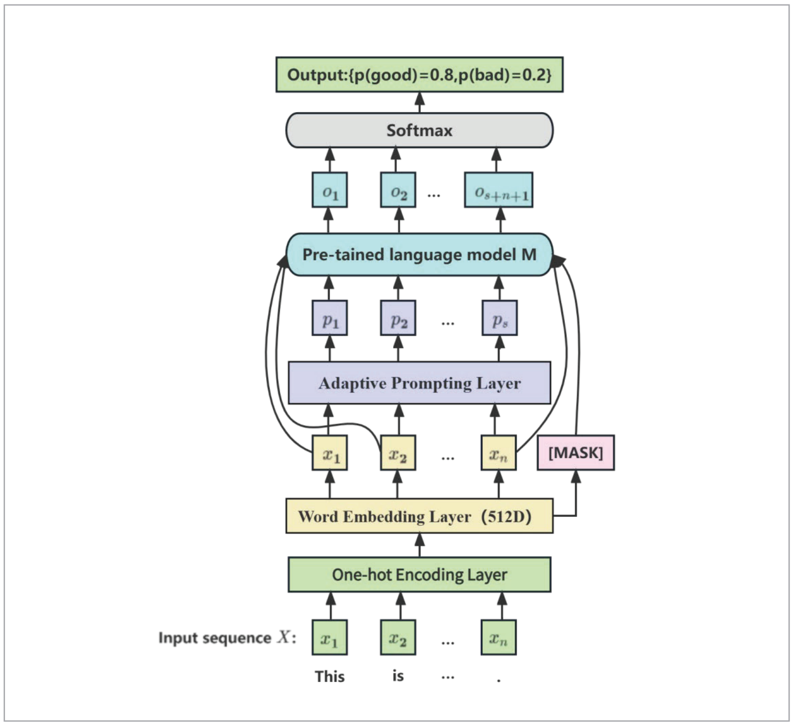
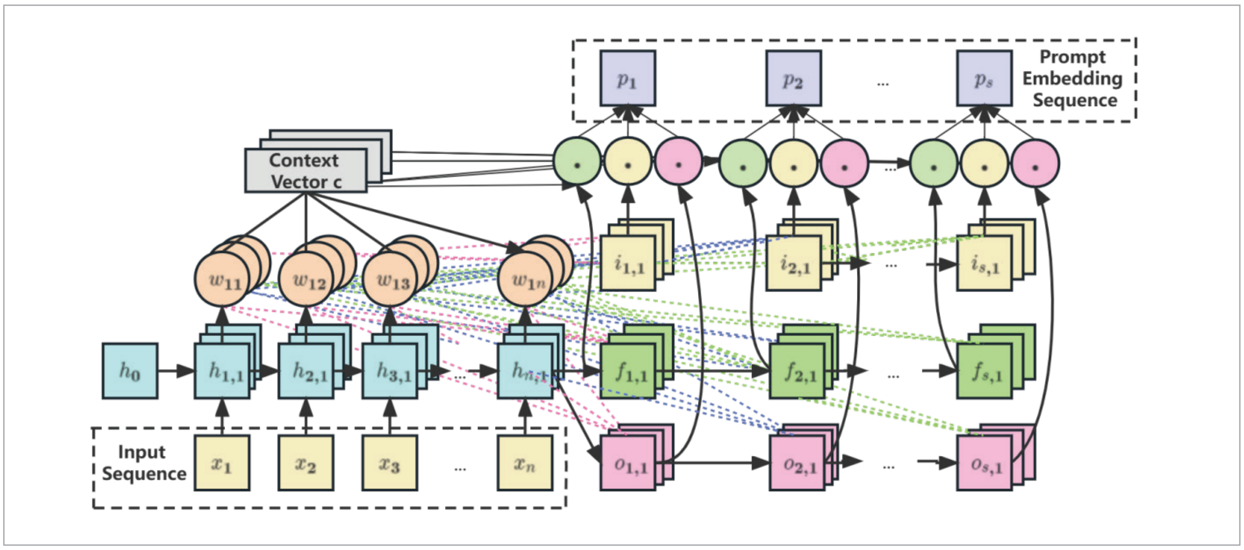

Output:{p(good)=0.8,p(bad)=0.2}
Softmax
o_1 o_2 ... o_{s+n+1}
Pre-tained language model M
p_1 p_2 ... p_s
Adaptive Prompting Layer
x_1 x_2 ... x_n [MASK]
Word Embedding Layer (512D)
One-hot Encoding Layer
Input sequence X:
This is ... .

Figure 5

Structure of the adaptive prompt learning layer



the context vector c. The right part of Figure 5 shows how the decoder uses the context vector c to decode and generate the outputs. In summary, the function of this module is to automatically create the prompt template based on the contents of the input text. Compared with traditional methods, our strategy fully utilizes the contextual information of the input X to automatically construct an adaptive training prompt template, which strengthens the correlation between the prompt information in the template and the input sequence X.

1 The encoder

We use a unidirectional LSTM encoder of k layers as our encoder. The input to this module is a set of 512-dimensional vectors of length $n:\{x_1, x_2 \dots x_n\}$, which is obtained after the input sequence X goes through the one-hot encoding layer and the word embedding layer. The output of the encoder is a set of 512-dimensional vectors of length $n:\{h_{1,k}, h_{2,k} \dots h_{n,k}\}$.

The inputs of the first layer are the words inputted at the current moment, the cell state, and the hidden state of the previous moment, while the outputs are the cell state and hidden state of the current moment. In the subsequent layers, the inputs are the hidden state of the previous layer at the current moment and the cell state and hidden state of the previous moment, while the outputs are the cell state and hidden state of the current moment. The detailed mathematical principles are shown in Equations (1)-(6) as follows:

$$i_{t,j} = \sigma\left(W_i\left[h_{t-1,j}, h_{t,j-1}\right]\right) \quad (1)$$

$$f_{t,j} = \sigma\left(W_f\left[h_{t-1,j}, h_{t,j-1}\right]\right) \quad (2)$$

$$o_{t,j} = \sigma\left(W_o\left[h_{t-1,j}, h_{t,j-1}\right]\right), \quad (3)$$

where $h_{t,j}$ is the hidden state at time t in layer j (special ones are when layer is $j=1, h_{t,1}=x_t$), $i_{t,j}, f_{t,j}, o_{t,j}$ stand for the state of the input gate, forgetting gate and output gate at time t in layer j, respectively. W_i, W_f, W_o represent the learnable parameters of the model. Then the cell state can be calculated as follows:

$$\tilde{C}_{t,j} = \tanh\left(W_C\left[h_{t-1,j}, h_{t,j-1}\right]\right) \quad (4)$$

$$C_{t,j} = f_{t,j} \odot C_{t-1,j} + i_{t,j} \odot \tilde{C}_{t,j} \quad (5)$$

$$h_{t,j} = o_{t,j} \odot \tanh\left(C_{t,j}\right). \quad (6)$$

$C_{t,j}$ indicates the cell state at time t in layer j, while $\tilde{C}_{t,j}$ indicates the candidate information state at time

t in layer j. σ and $\tanh$ represent the sigmoid and tanh activation functions, respectively. W_C represents the learnable parameters of the model. $\odot$ stands for element-wise multiplication.

2 The decoder

Our encoder is also a unidirectional LSTM encoder of k layers. The input to the encoder includes the id of the target template generated at the previous moment, the cell state, and hidden state of the previous moment. The output of the encoder includes the cell state and hidden state of the current moment. For the subsequent layers, the input includes the hidden state of the previous layer at the current moment, along with the cell state and hidden state of the previous moment. The output consists of the cell state and hidden state of the current moment. Subsequently, the hidden state of the last layer is utilized as the query for the attention mechanism. This helps in computing the contextual features of the current moment in the encoder sequence $\left\{h_{1,k}, h_{2,k} \ldots h_{n,k}\right\}$. Finally, the hidden state of the last layer is combined in various ways and fed into a linear layer. This generates the id of the prediction template for the next moment. The mathematical principles underlying these operations are presented in Equations (7)-(17):

$$i'_{t,j} = \sigma\left(W'_i\left[h'_{t-1,j}, h'_{t,j-1}, c_t\right]\right) \tag{7}$$

$$f'_{t,j} = \sigma\left(W'_f\left[h'_{t-1,j}, h'_{t,j-1}, c_t\right]\right) \tag{8}$$

$$o'_{t,j} = \sigma\left(W'_o\left[h'_{t-1,j}, h'_{t,j-1}, c_t\right]\right), \tag{9}$$

where $h'_{t,j}$ stands for the hidden state at time t in layer j (special ones are when layer is $j=1, h'_{t,1} = y_{t-1}$, $i'_{t,j}, f'_{t,j}, o'_{t,j}$ stand for the state of the input gate, forgetting gate and output gate at time t in layer j, respectively. W'_t, W'_f, W'_o represent the learnable parameters of the model. f represents the fully connected neural network. Then the cell state can be calculated as follows:

$$\tilde{C}'_{t,j} = \tanh\left(W'_C\left[h'_{t-1,j}, h'_{t,j-1}, c_t\right]\right) \tag{10}$$

$$C'_{t,j} = f'_{t,j} \odot C'_{t-1,j} + i'_{t,j} \odot \tilde{C}'_{t,j} \tag{11}$$

$$h'_{t,j} = o'_{t,j} \odot \tanh\left(C'_{t,j}\right), \tag{12}$$

where $\underset{\sim}{C}'_{t,j}$ indicates the cell state at time t in layer j, while $C_{t,j}$ indicates the candidate information state at time t in layer j. σ and $\tanh$ indicate the sigmoid and tanh activation functions, respectively. W'_C represent the learnable parameters of the model. $\odot$ stands for element-wise multiplication.

Finally, the output can be calculated as follows:

$$c_t = \sum_{i=1}^{n} w_{t,i} h_{ik} \tag{13}$$

$$w_{t,i} = \frac{\exp\left(score\left(h'_{tk}, h_{ik}\right)\right)}{\sum_{i=1}^{n} \exp\left(score\left(h'_{tk}, h_{ik}\right)\right)} \tag{14}$$

$$score\left(h'_{tk}, h_{tk}\right) = h_{ik}^T h'_{tk} \tag{15}$$

$$score\left(h'_{tk}, h_{tk}\right) = h_{ik}^T W_\alpha h'_{tk} \tag{16}$$

$$score\left(h'_{tk}, h_{tk}\right) = h_{ik}^T W_\alpha h'_{tk} \tag{16}$$

$$p_t = \arg\max\left(\begin{matrix} soft\max \\ \left(f\left(\begin{matrix} h'_{tk} + c_t, \\ h'_{tk} * c_t, h'_{tk} \odot c_t \end{matrix}\right)\right) \end{matrix}\right) \tag{17}$$

To capture the input details more effectively, we utilize the hidden state of the decoding layer as the query for the attention structure. This query enables the model to identify the relevant information in the encoder sequence $\left\{h_{1,k}, h_{2,k} \ldots h_{n,k}\right\}$ and calculate the contextual features of the current moment. Subsequently, the hidden state of the decoding layer is combined with that of the last layer in various ways and fed into the linear layer. This generates the id of the prediction template for the next moment.

3.3. Contrastive Learning Module

Overfitting is a common problem in few-shot learning due to the limited availability of data. To mitigate its impact, we incorporate the technique of R-Drop, which was proposed by Liang et al. [13] based on contrastive learning. This approach enables us to amplify the data while preserving the semantic information. By doing so, we can reduce overfitting and improve the generalization ability of the model. The mathematical analysis of this approach is presented below:

Given the training dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, our goal is to learn a model $P^W(x|y)$, where n represents the number of training samples, (x_i, y_i) is the labeled data pairs. x_i stands for the input data, while y_i corresponds to its corresponding label. In this task, x_i represents the sentence to be classified, and y_i represents the predicted class label of the model. We denote the probability distribution of the map function as $P^W(y|x)$, and indicate the Kull-Leibler (KL) divergence between the two distributions P_1 and P_2 as $D_{KL}(P_1 \| P_2)$. The goal of the method is to minimize the negative log-likelihood loss function, the details are shown as follows:

$$L_{nll} = \frac{1}{n}\sum_{i=1}^{n} -\log P^W(y_i|x_i). \tag{18}$$

In the training stage, we employ submodels that consist of random R-Drop units. However, in the inference stage, we use the full model without Dropout. Moreover, the submodels that arise from the randomly sampled Dropout units are dissimilar and lack constraints. Taking into account the randomness of the structure caused by Dropout and the above observations, we use the R-Drop method to regulate Dropout in the output predictions of the submodels.

During each training step, the input data x_i is fed twice through the network to obtain two distributions predicted by the model, which are recorded as $P_1^W(y_i|x_i)$ and $P_2^W(y_i|x_i)$, respectively. Since the Dropout operator drops random units in the model, the two forward transitions are based on two different sub-models, even though they are in the same model. The dropped units of each layer in the output path $P_1^W(y_i|x_i)$ are different from the correct path of the output distribution $P_2^W(y_i|x_i)$. Consequently, the distributions of $P_1^W(y_i|x_i)$ and $P_2^W(y_i|x_i)$ are different for the same input data pair.

Then, in each training step, the R-Drop method aims to minimize the bidirectional Kullback-Leibler (KL) divergence between the two output distributions for the same sample to regularize the prediction of the model. The formulas for this are as follows:

$$L_{KL}^{i} = \frac{1}{2}\begin{pmatrix} D_{KL}\left(P_1^W(y_i|x_i) \| P_2^W(y_i|x_i)\right) \\ +D_{KL}\left(P_2^W(y_i|x_i) \| P_1^W(y_i|x_i)\right)\end{pmatrix}. \tag{19}$$

Two positive transferred negative log-likelihood are used to learn the target L_{NLL}^{i}:

$$L_{NLL}^{i} = -\log P_1^W(y_i|x_i) - \log P_2^W(y_i|x_i) \tag{20}$$

The final training goal is to minimize the L^i of (x_i, y_i):

$$\begin{aligned} L^i &= L_{NLL}^{i} + \alpha \cdot L_{KL}^{i} \\ &= -\log P_1^W(y_i|x_i) - \log P_2^W(y_i|x_i) \\ &+ \frac{\alpha}{2}\begin{bmatrix} D_{KL}\left(P_1^W(y_i|x_i) \| P_2^W(y_i|x_i)\right) \\ +D_{KL}\left(P_2^W(y_i|x_i) \| P_1^W(y_i|x_i)\right)\end{bmatrix}, \end{aligned} \tag{21}$$

where α is the coefficient weight for controlling L_{KL}^{i}. This approach helps to further regularize the model space, leading to an improved generalization ability. By comparing Equation (18) with Equation (21), we can observe that during the training step, a KL-divergence loss L_{NLL}^{i} is added after performing two forward processing.

3.4. Training Process

The complete training process of the model can be divided into three steps as follows:

Step 1: Conduct the forward propagation process. Suppose there is a pre-trained model M, and the input sequence is a sentence X that have already been tokenized, and n stands for the length of the sentence X. The dimension of the word vector in the word embedding layer is 512. By sending X to the one-hot encoding layer, and then the word embedding layer, we obtain a set of 512-dimensional vectors with a length of n, denoted as $\{x_1, x_2 \ldots x_n\}$. Next, this set of 512-dimensional vectors is used as the input to the adaptive prompt layer proposed in this paper. The output is a set of 512-dimensional vectors of length s, denoted as $\{p_1, p_2 \ldots p_s\}$. Then, this set of 512 dimensional vectors $\{p_1, p_2 \ldots p_s\}$ of length s, together with $[MASK]$ and $\{x_1, x_2 \ldots x_n\}$ become the input of the pre-trained language model M (e.g., BERT) (where $[MASK]$ is a 512 dimensional vector of length 1 obtained after being processed by the word embedding matrix). The output is a set of 512 dimensional vectors with a length of $s+n+1$, denoted as $\{o_1, o_2 \ldots o_{s+n+1}\}$.

Step 2: To introduce implicit data enhancement in the training step, the Kullback-Leibler (KL) divergence is used to enhance the effectiveness of the model. Considering the randomness of dropout when passing through the network of the model, the data is passed and Step 1 is repeated each time, resulting in another set of 512-dimensional vectors of length $s+n+1$ denoted as $\left\{o^*_1, o^*_2 \dots o^*_{s+n+1}\right\}$.

Step 3: Do the backpropagation process by computing the negative log-likelihood for $\left\{o_1, o_2 \dots o_{s+n+1}\right\}$ and $\left\{o^*_1, o^*_2 \dots o^*_{s+n+1}\right\}$, respectively, and calculate the KL divergence between the two sets $\left\{o_1, o_2 \dots o_{s+n+1}\right\}$ and $\left\{o^*_1, o^*_2 \dots o^*_{s+n+1}\right\}$. By combining these two loss parts and performing backpropagation, we can update the learnable parameters in the model.

4. Study and Analysis of the Results

In this section, we present the datasets used for our experiments, as well as the comparative experiments conducted with other algorithms, and the migration experiments based on various fields, all of which are described in detail.

4.1. Datasets

We evaluated our model using the FewCLUE public datasets, specifically the EPRSTMT (E-commerce Product Review Dataset for Sentiment Analysis) task, which involves binary classification of sentiment analysis on e-commerce product reviews. The dataset was collected by ICIP Lab of Beijing Normal University and consists of a training set (160), validation set (160), public test set (610), test set (753), and unlabeled corpus set (19565). Since the training set is small with only 160 samples and the number of test data is more than 600, this presents a typical few-shot learning scenario. The ratio of labels in this dataset is Negative: Positive=86:74, indicating a balanced distribution, and implying that difficult samples may not exist. Based on Figure 6, the length of the input text is generally below 350 words, with only a few instances having more than 300 words, thus indicating that this task involves short text classification.

Other datasets we used in the migration experiment including the following: the social media public sentiment datasets (the data of reviews on Sina Weibo, including more than 100,000 data points, about 50,000 positive and negative comments each), the user comments data from a take-out platform (comment data collected by a delivery platform, with 4,000 positive comments and about 8,000 negative comments), the hotel comment dataset (including more than 7,000 hotel review data points, with over 5,000 positive and 2,000 negative comments), and the online shopping data of ten categories (books, tablets, mobile phones, fruit, water heaters, shampoo, Mengniu, hotels, computers, clothes, with a total of more than 60,000 reviews, about 30,000 positive and negative comments each).

Figure 6

Text length distribution graph of each piece of data (with label)

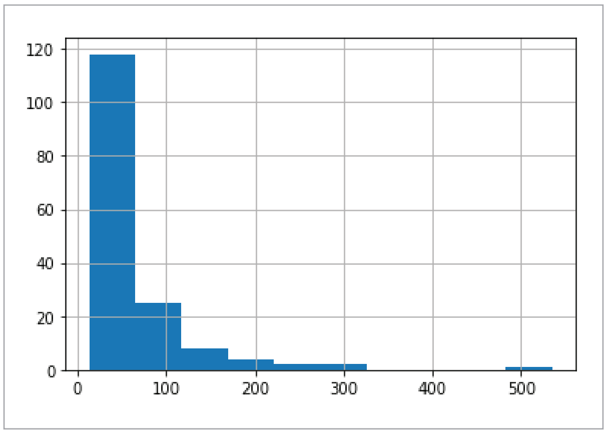


Table 1

Summary table of the large datasets used in migration experiment

Name of datasets	Number of data points	Number of positive comments	Number of negative comments
weibo_senti_100k	119,988	59,993	59,995
waimai_10k	11,987	4,000	7,987
ChnSentiCorp_htl_all	7,766	5,322	2,444
online_shopping_10_cats	58,274	27,228	31,046

4.2. Parameters Setting

In our experiment, we employed an A100 36G graphics card and the PyTorch 1.12 framework. To capture all the information of the input sentences, we set the maximum sentence length to twice the length of the digits in the dataset used in this paper. We used Albert-large as the pre-trained language model, which reduces the number of parameters of BERT while maintaining its performance and improving the efficiency of parameters [12]. The batch size was set to 10, and the output length of the adaptive prompt layer was 2 for Chinese and 4 for English, with α set to 4. The model employed the Adam optimizer and utilized different learning rates for different optimization methods.

4.3. Optimization Strategy

The model proposed in this study comprises of two trainable components: the parameters of the pre-trained model and the parameters of the dot-product attention in the adaptive prompt layer. On basis of this, our optimization method can be classified into two categories: the first involves fine-tuning all parameters (prompt + LM tuning). In this setting, there are prompt-related parameters that can be fine-tuned together with all or part of the parameters of the pre-trained models [17]. The second involves fine-tuning only the attention part (fixed LM prompt tuning). In this scenario, the additional prompt-relevant parameters are introduced besides the parameters of the pre-trained model, fixed-LM prompt tuning strategy updates only the parameters of the prompt using the supervision signal obtained from the downstream training samples while keeping the entire pre-trained LM unchanged [17].

4.4. Results

We show our experiment results in three parts according to our testing targets and parameters setting strategies.

4.4.1. Prompt + LM Tuning

In this section of our experiment, we employed the full model parameter adjustment method to evaluate the performance of our model on the eprstmt dataset from FewCLUE, which includes 32 training sets and about 600 test sets. In this setting, the fine-tuning of the pre-trained model plays a dominant role in the entire training process, while the attention part acts as an auxiliary component. Specifically, the manually-crafted prompt templates play a major role, and the adaptive prompt templates serve as a complement and enhancement to the manucally-crafted prompts. The learning rate is set to 1e-5. The results of this experiment are presented in Table 1.

Specifically, the zero-shot learning method uses only the prompt to construct the template, and then predicts through the pre-trained model without fine-tuning the parameters of the pre-trained model. We use the results of zero-shot learning to evaluate the quality of the hand-crafted template. In this case, it can be seen that the quality of the hand-crafted template has a greater impact on the HPL method and a smaller impact on the APRD method. The APRD method can also achieve higher accuracy when the hand-crafted template is not so good. On the other hand, the APRD model can outperform the HPL model when the manually-crafted prompt is good. Thus, this model can learn to adjust the weights of both hand-crafted and adaptive prompts to return better results. Overall, this finding confirms that APRD performs better than the HPL model.

Table 2
Accuracy of different strategies under different manual prompts in the eprstmt dataset

Prompt	Zero-shot Learning	HPL	APRD
_ _开心 (happy)	75.3%	83.6%	84.7%
_ _高兴 (glad)	69.9%	79.5%	84.4%
_ _不错 (good)	65.2%	80%	83.1%
_ _还行 (OK)	52.1%	77.4%	82.6%

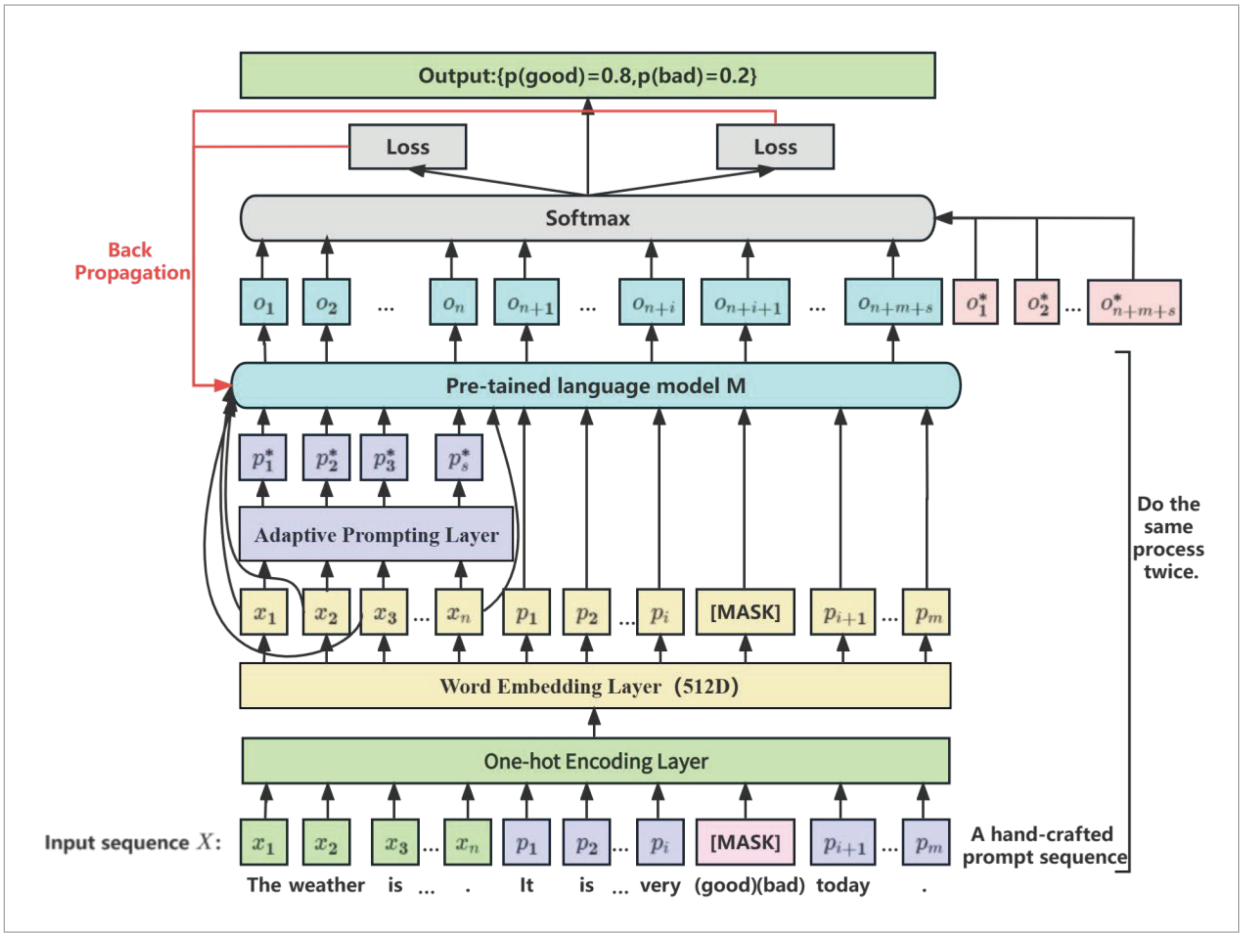
In order to examine the impact of the adaptive prompt component on the hand-crafted prompt method and compare its accuracy with that of the HPL method, we created a hybrid prompt learning model by combining the hand-crafted prompt component and the adaptive prompt component. The structure of the model is illustrated in Figure 7.

The hybrid prompt learning strategy combines both the word vector generated by the adaptive prompt layer and that generated by the hand-crafted prompt layer. The template is represented as a triple by the hybrid prompt embedding layer:

$$\left\langle X, P, p^{*}\right\rangle = \left\{e\left(p_1\right), e\left(p_2\right)\ldots e\left(\left[MASK\right]\right), e\left(p_{i+1}\right)\ldots e\left(p_m\right), p_1^{*}, p_2^{*}\ldots e\left(x_1\right), e\left(x_2\right)\ldots e\left(x_n\right)\right\},$$

where P represents the manually-crafted prompt template, p^* represents the sequence of the embedding vector of the adaptive prompt, and X stands for the sequence of the input text. The set of the final prediction results of the model is $y=\left\{p(i), i \in W\right\}$, which is calculated by the pre-trained model. In this strategy, both the adaptive and hand-crafted prompt

Figure 7
Structure of the hybrid prompt model


Output:{p(good)=0.8,p(bad)=0.2}
Loss
Loss
Softmax
Back Propagation
o_1 o_2 ... o_n o_{n+1} ... o_{n+i} o_{n+i+1} ... o_{n+m+s} o_1^* o_2^* ... o_{n+m+s}^*
Pre-tained language model M
p_1^* p_2^* p_3^* p_s^*
Adaptive Prompting Layer
x_1 x_2 x_3 ... x_n p_1 p_2 ... p_i [MASK] p_{i+1} ... p_m
Do the same process twice.
Word Embedding Layer (512D)
One-hot Encoding Layer
Input sequence X:
A hand-crafted prompt sequence
The weather is ... . It is ... very (good)(bad) today .

parts influence the final outcome. The results in this section show that this model can obtain output results by learning to adjust the weights of P and p^*. Therefore, theoretically, the model has the advantages of both hand-crafted prompt and adaptive prompts. The effectiveness of the model can be further improved when a "good" hand-crafted prompt template is found. The adaptive prompt module can generate good prompt templates even if a "good" hand-crafted prompt template is not found. This proves the effect of the adaptive prompt module on the model.

4.4.2. Fixed LM Prompt Tuning

To further test the ability of the dot-product attention part in the APRD method, we conducted an experiment where we removed the hand-crafted prompt method and used only the attention part to generate prompts. The parameters of the pre-trained language model were frozen in this experiment. The goal of the attention structure was to learn the embedding representation of the adaptive prompt in the language model, so as to make it perform like the real text sequence through the embedding layer.

We conducted experiments on both small sample datasets (EPRSTMT of FewCLUE datasets) and large-scale datasets (microblog data). The results on the large-scale dataset showed that the model achieved an accuracy of over 93%, while on the small sample dataset, the accuracy was only around 66%.

Our experiments demonstrate that the model performs well on large-scale datasets, but shows poor performance on small sample datasets. We hypothesize that in the scenario of large datasets, due to the availability of sufficient samples, the model can learn the adaptive prompt and its embedding representation in the pre-trained model through the dot-product attention structure. However, in the case of limited samples, the attention structure struggles to learn both parts simultaneously, leading to over-fitting.

In conclusion, our experiments suggest that the embedded representation can only be learned on large-scale datasets. Our findings also demonstrate that the model has the ability to learn an adaptive prompt with sufficient samples. Therefore, in order to further verify the ability of the dot-product attention structure to learn the embedded representation of the pre-trained model and the generality of the adaptive prompt, we conducted a migration experiment.

4.4.3. Migration Experiment

Sentiment analysis spans across various domains such as delivery, reviews on social media, e-commerce, and catering. Although there may be diversities among these domains, sentiment classification is usually required. The main reason why past models cannot be directly used across domains is due to the differences in words and language structures used to express sentiment, leading to variations in parameters in the word vector and fully connected layers. Hence, an effective automated prompt construction method should be capable of learning adaptive prompts in general fields and deliver satisfactory results in unknown domains.

In this part of the experiment, we conducted a mixed-data experiment where we combined sentiment analysis datasets in the online shopping field, the hotel field, themicroblog field, and the takeout field for the training set. We used EPRSTMT dataset as the test set. The learning rate was set at 1e-6, and the result is presented in Figures 8-9.

It can be observed that the proposed model can successfully learn to construct general adaptive prompts for mixed fields and performs well in other fields with an average accuracy of 89.2%, which is significantly higher than the results of methods employed in other sections on the FewCLUE datasets, as presented in Table 3.

Figure 8
Accuracy of the datasets of different fields

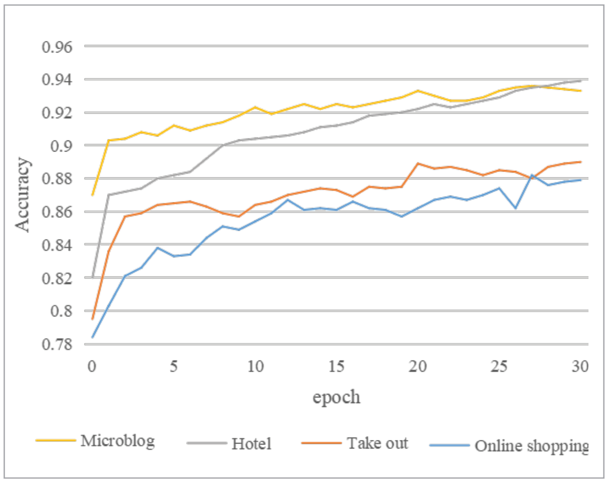


Figure 9
Accuracy of the test dataset

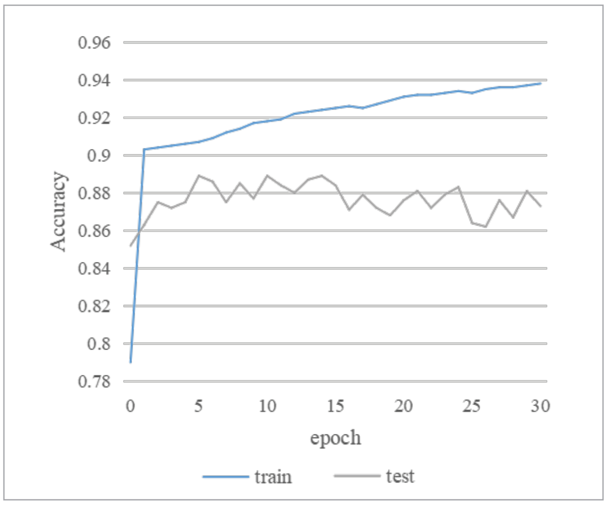


Table 3
The main results for different methods on FewCLUE datasets

Method	Fine-Tuning	PET	P-tuning	LM-BFF	EFL	APRD
Accuracy	65.4%	86.7%	88.3%	85.6%	84.9%	89.2%

5. Conclusion

In this paper, we first make a conclusion of the limitations of existing main prompt learning strategies and few-shot learning methods, then propose a new sentiment analysis method by the combination of adaptive prompt learning and contrastive learning. The advantages of our work are summarized as follows:

1 We successfully integrate hand-crafted prompts and adaptive prompts into a single model.

2 We introduce a dot-product attention structure and leverage contextual information to automatically generate adaptive prompts.

3 The model APRD we proposed can learn to create general adaptive prompt by doing training on large sentiment analysis datasets.

4 However, the model performs well only on large-scale datasets, as the dot-product attention structure fails to learn the adaptive prompt and the embedded representation in the pre-trained model simultaneously, leading to over-fitting. To improve the performance of the model, our future research will focus on the following areas: 1) Develop a more efficient parameter fine-tuning method based on the pre-trained prompt; 2) Extend our method to other NLP tasks, such as other types of text classification and machine reading comprehension; 3) Enhance the ability of the model to perform multi-language sentiment analysis tasks, particularly on English sentiment analysis datasets; 4) Try to make refinements and explore the possibility of using this method in fine-grained or cross-grained sentiment analysis tasks.

Data Sharing Agreement

The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, author-ship, and/or publication of this article.

Funding

This work is supported by National Natural Science Foundation of China (12201185), the Kaifeng Science and Technology R&D Project No.2201009, Henan Province Science and Technology R&D Project (232102210031, 232102210029, and 222102210034), and the Postgraduate Education Reform and Quality Improvement Project of Henan Province (YIS2022ID26).

References

1. Brown, T. B., Mann, B., Ryder, N., et al. Language Models are Few-shot Learners. International Conference on Neural Information Processing Systems, 2020, 1877-1901.
2. Chen, J., Yu, J., Zhao, S., Zhang, Y. User's Review Habits Enhanced Hierarchical Neural Network for Document-Level Sentiment Classification. Neural Processing Letters, 2021, 53(3), 2095-2111. https://doi.org/10.1007/s11063-021-10423-y
3. Chen, L. C., Lee, C. M., Chen, M. Y. Exploration of Social Media for Sentiment Analysis Using Deep Learning. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2020, 24(11), 8187-8197. https://doi.org/10.1007/s00500-019-04402-8
4. Devlin, J., Chang, M W., Lee, K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Association for Computational Linguistics: Human Language Technologies, 2019, (1), 4171-4186. https://doi.org/10.18653/v1/N19-1423
5. Ding, N., Hu, S., Zhao, W., Chen, Y., Liu, Z., Zheng, H. T., Sun, M. OpenPrompt: An Open-source Framework for Prompt-learning. Association for Computational Linguistics: System Demonstrations, 2022, 105-113. https://doi.org/10.18653/v1/2022.acl-demo.10
6. Ding, N., Chen, Y., Han, X., Xu, G., Xie, P., Zheng, H. T., Liu, Z., Li, J., Kim, H. G. Prompt-learning for Fine-grained Entity Typing. ArXiv, 2021, 2108, 10604. https://doi.org/10.18653/v1/2022.findings-emnlp.512
7. Fang, Y., Sun, J., Han, B. Research on Text Sentiment Analysis Method Based on BERT. Information Technology and Informatization, 2020, (2), 108-111.
8. Gao, T., Fisch, A., Danqi, C. Making Pre-trained Language Models Better Few-shot Learners. ArXiv: 2012. 15723, 2021. https://doi.org/10.18653/v1/2021.acl-long.295
9. Heikal, M., Torki, M., El-Makky, N. Sentiment Analysis of Arabic Tweets Using Deep Learning. Proceedings of the 4th Annual International Conference on Arabic

Computational Linguistics. 2018, 114-122. https://doi.org/10.1016/j.procs.2018.10.466

10. Hu, S., Ding, N., Wang, H., Liu, Z., Li, J., Wu, W., Sun, M. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. Association for Computational Linguistics, 2022, 1, 2225-2240. https://doi.org/10.18653/v1/2022.acl-long.158
11. Lafferty, J., Mccallum, A., Pereira, F. C. N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. The Eighteenth International Conference on Machine Learning. 2001, 282-289.
12. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. ArXiv, 2019, 1909, 11942.
13. Le, Q V., Mikolov, T. Distributed Representations of Sentences and Documents. The 31st International Conference on International Conference on Machine Learning. 2014, 2(32), 118-1196.
14. Li, D., Sun, L., Xu, X., Wang, Z., Zhang, J., Du, W. BLSTM and CNN Stacking Architecture for Speech Emotion Recognition. Neural Processing Letters. 2021, 53, 4097-4115. https://doi.org/10.1007/s11063-021-10581-z
15. Li, G., Lin, Z., Wang, H., Wei, X. A Discriminative Approach to Sentiment Classification. Neural Process Letters, 2020, 51, 749-758. https://doi.org/10.1007/s11063-019-10108-7
16. Liang, X., Wu, L., Li, J., Wang, W., et al. R-drop: Regularized Dropout for Neural Networks. Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, 2021, 10890-10905.
17. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys, 2021, 55(9), 1-35. https://doi.org/10.1145/3560815
18. Liu, S., Feng, X. Text Sentiment Analysis Based on BERT. Journal of Information Security Research, 2020, 6(3), 220-227.
19. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J. GPT Understands, too. ArXiv, 2021, 2103, 10385.
20. Pan, D. H., Yuan, J. L., Li, L., Sheng, D. Deep Neural Network-Based Classification Model for Sentiment Analysis. 6th International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC), 2019, 1-4. https://doi.org/10.1109/BESC48373.2019.8963171
21. Pang, B., Lee, L. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2008, 2(1-2), 130-135. https://doi.org/10.1561/1500000011
22. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. Deep Contextualized Word Representations. Association for Computational Linguistics: Human Language Technologies. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, 1, 2227-2237. https://doi.org/10.18653/v1/N18-1202
23. Sadr, H., Pedram, M. M., Teshnehlab, M. A Robust Sentiment Analysis Method Based on Sequential Combination of Convolutional and Recursive Neural Networks. Neural Process Letters, 2019, 50, 2745-2761. https://doi.org/10.1007/s11063-019-10049-1
24. Schick, T., Schütze, H. Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference. Association for Computational Linguistics, 2021, Main Volume, 255-269. https://doi.org/10.18653/v1/2021.eacl-main.20
25. Schick, T., Schütze, H. Exploiting Cloze-questions for Few-shot Text Classification and Natural Language Inference. Association for Computational Linguistics: 2021, Main Volume, 255-269. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.20
26. Schick, T., Schütze, H. Few-shot Text Generation with Natural Language Instructions. Conference on Empirical Methods in Natural Language Processing, 2021, Main Volume, 390-402. https://doi.org/10.18653/v1/2021.emnlp-main.32
27. Sun, M., Yang, L., Zhengfei, Z., Tao, Q. Sentiment Analysis Based on BGRU and Self-Attention Mechanism. Journal of Jianghan University (Natural Science Edition), 2020, 48(4), 80-89.
28. Sun, X., Peng, X., Hu, M., Ren, F. Extended Multi-modality Features and Deep Learning Based Microblog Short Text Sentiment Analysis. Journal of Electronics & Information Technology, 2017, 39(9), 2048-2055.
29. Tong, R. M. An Operational System for Detecting and Tracking Opinions in On-line Discussion. ACM Sigir Workshop on Operational Text Classification, 2001, 1-6.
30. Yadav, A., Vishwakarma, D. K. Sentiment Analysis Using Deep Learning Architectures: A Review. Artificial Intelligence Review, 2020, 53, 4335-4385. https://doi.org/10.1007/s10462-019-09794-5
31. Ye, H., Zhang, N., Deng, S., Chen, X., Chen, H., Xiong, F., Chen, X., Chen, H. Ontology-enhanced Prompt-tuning for Few-shot Learning. Proceedings of the ACM Web Conference 2022. 2022, 778-787. https://doi.org/10.1145/3485447.3511921